Incremental Placement for Timing Optimization

Wonjoon Choi wjchoi@ece.umn.edu
Kia Bazargan kia@ece.umn.edu
ECE Department , University of Minnesota, Minneapolis, MN, USA

Abstract

An incremental timing driven placement algorithm is presented. We introduce a fast path-based analytical approach for timing improvement. Our method achieves timing optimization by reducing the enclosing bounding boxes of selected nets on critical paths. Furthermore, this technique tries to minimize modifications to the initial placement while improving the delay of the circuit incrementally. Two contributions of this work are 1) efficient conversion of a path-based timing minimization problem to a geometric net-constraint problem and 2) minimal modification of a placement to improve timing. Our technique can take an initial placement from any algorithm and improve timing iteratively. The experiments show that the proposed approach is very efficient.

1. Introduction

Timing driven placement is an important step in physical design of integrated circuits. In high-speed circuits, a significant portion of timing minimization is performed in the placement stage in the form of wire length minimization, followed by further optimization techniques such as buffer insertion either during placement or routing. In complex designs of today, various qualities such as delay, power consumption, heat profile and area are optimized in an intricate process that can strike a balance between these metrics. Multiple iterations might be needed for timing closure. Given the complexities of the designs and tight time-to-market constraints, it is highly desired to have a placement algorithm that 1) needs fewer iterations of the whole optimization cycle, and 2) in minimizing a given metric such as delay, disturbs the current placement minimally so that other metrics such as heat profile of the design do not change dramatically. For these reasons, incremental placement methods that focus on the most critical paths in the design would be very useful in design convergence.

Timing optimization during placement has been an important research topic especially since we entered the deep sub-micron era. Research efforts on this problem can be classified into two categories: path-based and net-based approaches. Path-based techniques enumerate some critical paths and try to optimize them, hence decreasing the overall delay of the circuit. Edge-based methods assign each edge with a "*criticality*" number that indicates how the edge contributes to the delay of the circuit based on the paths that pass through it. Path-based methods could suffer from large runtimes because of the potentially exponential number of paths in a circuit. On the other hand, edge-based methods suffer from the fact that they lose the big picture: they no longer see paths, but only edges. Hence, the efforts to minimize wire lengths corresponding to critical edges might not be as harmonious as the path-based methods.

Among works on timing driven placement, [1] puts higher weights on timing critical nets in the framework of force directed placement. Difficulty with this approach is to determine how much weight should be increased and which particular nets along the timing path should be weighted more, i.e., the slack assignment problem. Delay from interconnection could be estimated by the size of net bounding box during placement. Direct control on the size of bounding box was presented in [2][3]. They used linear programming to ensure all nets meet their bounding box constraints. The work in [4] shows a post-routing timing optimization technique with long runtimes. We present a timing optimization technique applied during placement that uses "net constraints" to reduce circuit delay. "Net constraint" is the maximum size of the bounding box of a net.

The net constraint generation process should consider the path. If one critical path consists of a few net segments, more constraint should be put on individual nets on the path compared to a case in which the path consists of many edges. In the latter case, small contractions of the nets can add up to a significant delay decrease, a luxury that the former case lacks. In the past, heuristic methods (or user input) have been used to decide what constraints to put on different nets. The quality of net constraint generation is crucial to the effectiveness of a net-constraint-based timing driven placement [3]. Here, we propose analytic formulations to generate net constraints to reduce circuit delay in such a way that placement disturbance is minimum.

2. Problem Formulation

The delay of a net in the placement step is usually estimated by the half perimeter of the net's bounding box, which is the smallest rectangle enclosing all pins connected by the net. The maximum delay on any path from a primary input pin (or memory element output) to a primary output pin (or a memory element input) determines the delay of the circuit. If the delay is more than the target delay of the design, the placement is invalid and should be further optimized.

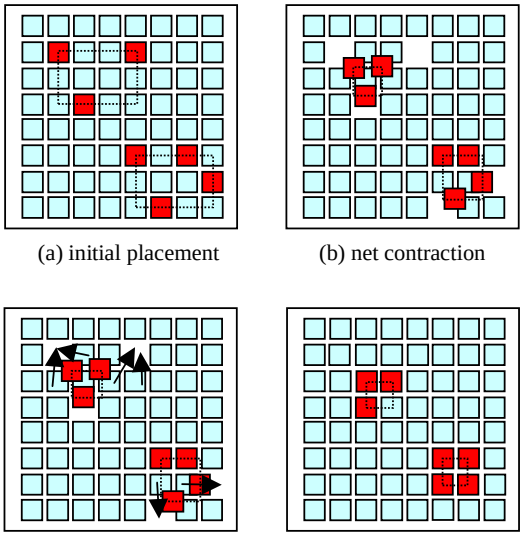
If the initial placement misses the target delay goal by a small amount, it has high chances of meeting the timing goal by incremental changes. Figure 1 shows a simple illustration of our proposed flow. The input to our flow is the placement generated by any placement method (*e.g.*, Dragon). Then, critical paths are identified as part (a) next, the bounding boxes are contracted as in part (b) and overlaps are removed as in part (c). Finally, the new placement with better timing is obtained as in part (d). Figure 1 illustrates one iteration (a-d) of our iterative algorithm.

To improve timing, bounding boxes on the critical paths should be reduced. The number of paths could increase exponentially as the circuit size grows. So, to keep the problem size manageable, we use *intermediate target timing* and consider a certain number of critical paths and optimize placement for

these paths. Target timing can be achieved by iteratively reducing the incremental target timing. Another benefit of this iterative approach is that we can handle critical paths dynamically. If only critical paths (i.e., those currently violating delay bound) are considered during the whole optimization process, other non-critical paths tend to become critical as we move cells around and change bounding boxes of other nets. An iteratively incremental approach can handle those new critical paths in the following iterations.

Two different approaches to the timing optimization are given here. The first approach uses net bounding boxes as intermediate variables in the timing optimization process: path negative slacks are translated to net contractions in one linear programming formulation, followed by another problem that achieves net contractions by moving cells. The second approach uses a single formulation to move cells directly in order to meet timing constraints, while minimizing cell movements. Even though the second approach is more intuitive, but it results in far worse runtimes and is presented in the Appendix. Sections 2 and 3 describe the first approach.

The details of our approach are presented in subsequent sections in the following order. Section 3 shows how to identify nets whose bounding boxes are to be reduced. In Section 4, we show how cells are moved to other places with minimum impact on the initial placement. The final step of the proposed technique is legalization and is explained in Section 5. Experimental results are in Section 6, followed by concluding remarks in Section 7.



(a) initial placement (b) net contraction

(c) legalization (d) after one iteration

Figure 1. Flow of the incremental placement

3. Net Constraint Generation

In this section, we describe our incremental timing-optimization technique. Our method is two-tier. In the first step, we try distribute "net constraints" on net bounding boxes to reduce delay on critical paths. The next section describes how we translate bounding box changes to individual cell movements.

Distributing negative slack of paths to individual nets has to be done carefully. If many critical paths share some nets, timing on those critical paths can be efficiently improved by reducing the bounding-boxes of the shared nets. This strategy aims to minimize the perturbation to the placement. The amount of reduction of bounding boxes can be determined analytically using linear programming.

For a placement with a delay $d_{initial}$, and for incremental target timing, d_{it} $(= d_{initial} - \Delta t)$, all paths that violate the incremental target timing are identified. If there are too many paths violating this d_{it}, only K paths are used and d_{it} is adjusted accordingly (in the experiment, K is set to 2000). For a path $path_i$ with path delay d_i, the amount of delay that should be decreased along $path_i$ is $(d_i - d_{it})$. The objective of the linear program is to minimize the net changes in order to minimize the impact on the placement. In the following expression, Δd_{ij} (> 0) is the decrease of wire delay on the j*th* net on $path_i$. Consider k $(\leq K)$ most critical paths, whose delays are more than d_{it}. To achieve the target timing, the following constraints should be met:

$$\begin{aligned} &path_1: \Delta d_{11} + \Delta d_{12} + \ \ldots + \Delta d_{1|E1|} \geq d_1 - d_{it} \\ &path_2: \Delta d_{21} + \Delta d_{22} + \ \ldots + \Delta d_{2|E2|} \geq d_2 - d_{it} \\ &\quad \ldots \\ &\mathrm{path_k}: \Delta \mathrm{d_{k1}} + \Delta \mathrm{d_{k2}} + \ldots + \Delta \mathrm{d_{k|Ek|}} \geq \mathrm{d_k} - \mathrm{d_{it}} \end{aligned} \quad (1)$$

Here $|Ei|$ is the number of nets on $path_i$. The delay reduction, Δd_{ie} is translated to bounding box changes, for a given delay model D, as follows.

$$\Delta d_{ij} = \frac{\partial D}{\partial B_{ijx}} \Delta B_{ijx} + \frac{\partial D}{\partial B_{ijy}} \Delta B_{ijy} \quad (2)$$

ΔB_{ijx} and ΔB_{ijy} are the reductions in the bounding box size of net_j on $path_i$ in the x- and y-direction respectively. Now (1) is rewritten as expressions of bounding boxes, as shown in (3), and used as a constraint in the linear programming formulation.

$$\begin{aligned} path_i: \ &\frac{\partial D}{\partial B_{i1x}} \Delta B_{i1x} + \frac{\partial D}{\partial B_{i1y}} \Delta B_{i1y} + \ldots \\ &+ \frac{\partial D}{\partial B_{iex}} \Delta B_{iex} + \frac{\partial D}{\partial B_{iey}} \Delta B_{iey} \geq d_i - d_{it} \end{aligned} \quad (3)$$

The objective of the linear programming is

$$\text{Min: } \sum_{i,j} (\Delta B_{ijx} + \Delta B_{ijy})^* \quad (4)$$

In (4), we construct the objective function in a way that each critical net appears only once, which is indicated with (*). As a result, this objective function will give higher priority to reducing the bounding boxes of the shared nets because they appear in more (3) inequalities.

To minimize the changes to the placement, each ΔB_{ijx} and ΔB_{ijy} is bounded to small values, as follows.

$$\begin{aligned} &\Delta B_{ijx} \leq p\, B_{ijx}, \\ &\Delta B_{ijy} \leq p\, B_{ijy} \\ &for\ 0 < p < 1 \end{aligned} \quad (5)$$

B_{ijx} and B_{ijy} are the size of the bounding box of net j of $path_i$ from the current placement. To decide the minimum p in (5), we

try to solve the linear program using (3), (4) and (5) with a small *p*. If no feasible solution is obtained, we increase it by a small amount (in the experiment, 5%). Intuitively, large ΔB_{ij} values are likely to change the current placement significantly and may cause more non-critical paths to become critical. Hence we opt for small *p*. By forcing all ΔB's as non-negative reduction, no net bounding boxes on critical paths will be expanded. The output of the linear program will be ΔB_{ijx} and ΔB_{ijy} values that have to be enforced to meet the incremental timing goal.

4. Contraction of the Critical Nets with Minimal Placement Changes

Using these tighter bounding boxes from the linear program in Section 3, cells are moved to other positions to meet the net constraints (i.e., bounding box changes are translated into cell movements). The way that cells are moved to other positions should be done with minimum perturbation to the current placement. This problem can be modeled as an optimization problem, whose objective is minimum placement perturbation with constraints that all pair of connected bounding boxes by some shared nodes should be kept as linked.

$$Min: \sum_{i \in N_c} (Xc_{net_i}^{old} - Xc_{net_i}^{new})^2$$

such that.

$$\left| Xc_{net_i}^{new} - Xc_{net_j}^{new} \right| < 0.5 * (Bx_{net_i}^{new} + Bx_{net_j}^{new}) \forall_{i,j \in N_c} \quad (5)$$

Here, Xc_{net_i} is the x-coordinate of the center of net_i bounding box and superscript *old* and *new* represent the current known *Xc* value and the unknown new center point respectively. N_c represents the nets on critical paths. And, $Bx_{net_i}^{new}$ represents the reduced bounding box of net_i in the x-direction. This objective function tends to keep the new placement as close to the old one as possible and to maximize the overlap between the new spans of net bounding boxes with their previous spans in the old placement.

By (5), the coordinate of each bounding boxes are determined. Now the location and size of each bounding boxes are known. So, critical cells can be moved to new location. The same idea is applied to the y-direction.

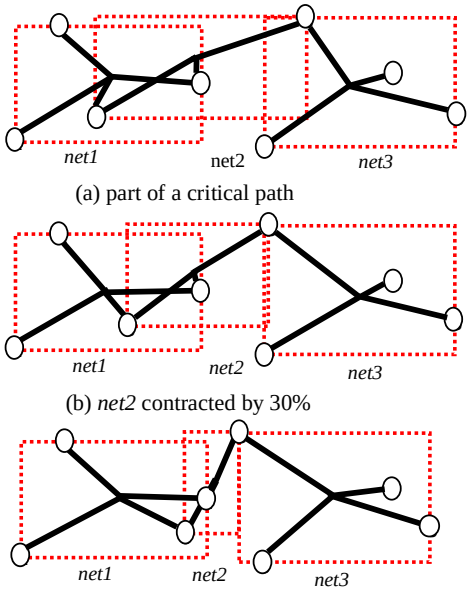
To better illustrate this idea, consider the example of Figure 2. Net_1 connects $cell_1$, $cell_2$, $cell_3$ and $cell_4$, net_2 connects $cell_3$, $cell_4$ and $cell_5$ and net_3 connects $cell_5$, $cell_6$, $cell_7$ and $cell_8$. Assume that the solution to the linear equations from Section 3 requires that the horizontal length of the bounding box of net_2 be reduced by 30%, with no reductions on net_1 and net_3. The solution would be something like part (b) in Figure 2. In this example, there are overlaps between net_1 and net_2 and between net_2 and net_3.

The solution would require only $cell_3$ and $cell_5$ to move toward the center of net_2. But if net_2 is to be reduced by 60% in the horizontal direction, the net contraction will result in part (c) in Figure 2. In this case the centers of net_2 and net_3 will move as a result of displacing $cell_3$, $cell_5$ and $cell_8$.

This technique reduces those critical nets at the expense of other non-critical nets. Once cells are moved to other places using the above technique, some overlaps are created. Hence, after the net contraction phase, an overlap resolving step should be followed.

5. Overlap removal

By contracting critical nets, overlaps are created. A simple technique can be applied to remove overlaps. We applied a simple bisection technique. First, after sorting cells by horizontal



(a) part of a critical path

(b) *net2* contracted by 30%

(c) net_2 contracted by 60%

Figure 2. Net contraction

(vertical) coordinates, cells are assigned to two child bins, which are created by a vertical (horizontal) cut of the placement area. This is followed by the same bisection placement in alternating horizontal/vertical directions until all overlaps are removed. In our experiment, during this process, white space was actively utilized to keep the total wire length increase small. If initial placement has uniformly distributed white space of 10%, we relaxed the minimum white space of each child region to 5%. The overlap removal method is fast because the only computation involved is sorting of the cells by their coordinates.

6. Experimental Results

We have implemented the proposed analytic approach for timing optimization and experimented on a Linux machine with Intel Pentium 930 MHz processor with 512 MB memory. We used lp_solver [5] to solve the linear program and OOQP [6] to solve the quadratic program. We ran our implementation for 4 test circuits from [7] and 4 from [8]. The same wire delay model as [7] and [8] were used respectively.

As input to our approach, we got an initial placement by running the placer of [7] and [8], followed by iterations of our proposed algorithm (either 5 iterations or until wire length increase was more than 10%). We set *incremental target timing* as follows and the used number of critical paths was 2000.

$$t_{it} = (t_{current} - t_{noWireDelay}) \times 0.8 + t_{noWireDelay}$$

The results are shown in Table 1. As we continue iterating the proposed optimization, we observed a trade-off curve, i.e. the total wire length generally increases and circuit timing improves.

An example of the trade-off curve is shown in Figure 3. In the case of placements generated by Dragon, our approach resulted in more than 10% increase in wire length in its first iteration, so only one iteration was used. Runtimes reported in the "our runtime" column are the total runtimes of all iterations.

7. Conclusion

In this paper, we proposed an analytic incremental placement technique to improve circuit timing. Our approach uses an optimization technique to minimize the perturbation on the placement. Experimental results show that the changes are small while achieving timing improvement in very short runtimes.

References

[1] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning", *ACM/IEEE Design Automation Conference*, 1998.

[2] B. Halpin, C. Y. roger Chen, and Naresh Sehgal, "Timing Driven Placement using Physical Net Constraints", *Design Automation Conference*, pp.780-783. IEEE/ACM, June 2001.

[3] K. Rajagopal et al., "Timing Driven Force Directed placement with Physical Net Constraints", *International Symposium on Physical Design*, pp 60-66. IEEE/ACM, April 2003.

[4] A. H. Ajami and M. Pedram, "Post-Layout Timing-Driven Cell Placement Using an Accurate Net Length Model with Movable Steiner Points", *ASPDAC*, 2001

[5] ftp://ftp.es.ele.tue.nl/pub/lp_solve/. Information and Communication Systems group at the Electrical Engineering department of the Eindhoven University of Technology, 1998.

[6] http://www.cs.wisc.edu/~swright/ooqp/. University of Chicago, 2001

[7] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis, "Multi-objective Circuit Partitioning for Cutsize and Path-based Delay Minimization," *ICCAD*, 2002

[8] X. Yang, B.K. Choi, and M. Sarrafzadeh, "Timing-Driven Placement using Design Hierarchy Guided Constraint Generation", *ICCAD*, 2002.

Appendix

Formulation using cell location directly:

$$\min: \sum_{i\in C_c}\left|\left|x_i^{new}-x_i^{old}\right|+\left|y_i^{new}-y_i^{old}\right|\right|+\alpha\sum_{j\in N_c}(x_{bj}^{r}-x_{bj}^{l}+y_{bj}^{t}-y_{bj}^{b})$$

such that

$$x_{bi}^{l} \le x_{bi}^{r},\ y_{bi}^{b} \le y_{bi}^{t} \quad \forall i \in N_c \qquad \text{(a1)}$$

$$x_k \le x_{bi}^{r},\ x_k \ge x_{bi}^{l},\ y_k \le y_{bi}^{t},\ y_k \ge y_{bi}^{b} \quad \forall k \in net_i,\ \forall i \in N_c \qquad \text{(a2)}$$

for path$_i$:

$$\sum_{k=1}^{k=|E|}\left\{\frac{\partial D}{\partial B_{ikx}}\left[B_{ikx}^{old}-(x_{bk}^{r}-x_{bk}^{l})\right]+\frac{\partial D}{\partial B_{iky}}\left[B_{iky}^{old}-(y_{bk}^{t}-y_{bk}^{b})\right]\right\} \le d_i - d_{it} \qquad \text{(a3)}$$

A linear solver can solve this formulation. To get rid of the absolute values in the objective function, we can introduce new variables and constraints. We can replce $|Z|$ by $Z_{positive}$ in the objective function. Constraints $-Z_{positive} \le Z \le Z_{positive}$ have to be added.

All notations in (a1)-(a3) are as follows.

$x_i^{new}, x_{old}^{i}, y_{new}^{i}, y_{old}^{i}$: new and old x,y-coordinate of $cell_i$

$x_{bj}^{r}, x_{bj}^{l}, y_{bj}^{t}, y_{bj}^{b}$: right, left, top and bottom coordinates of bounding-box$_j$

C_c, N_c: cells and nets on critical paths, respectively.

(a1) and (a2) represent the requirement of bounding box for the cells of a net. (a3) represents the delay requirement. Our experiments showed that this formulation gives a little worse quality (for the same delay, wirelength increase is worse) and very long run-time (2m vs 4h). This result is shown in Figure 3.

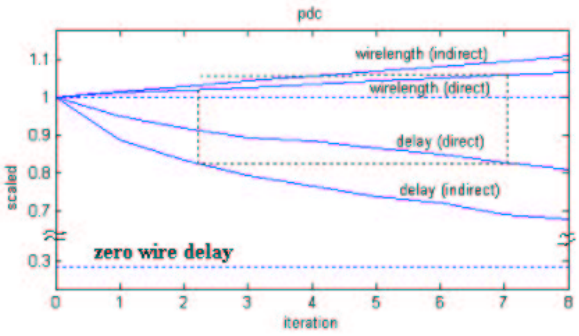


Figure 3. delay and wirelength change ("direct" formulation in appendix, "indirect" formulation in Sections 3 and 4)

Table 1. Experimental results

circuit	Cells	nets	zero-wire delay (ns)	initial delay	runtime [8]*	runtime [7]**	Our runtime	our delay	wirelength increase
matrix	3,083	3,200	4.93	4.99	5 m		1 m	no improve	NA
VP2	8,714	8,789	5.59	5.67	19 m		1 m	no improve	NA
32-MAC	8,902	9,115	1.96	2.88	26 m		2 m	2.77	10.5 %
64-MAC	25,616	26,017	2.34	6.00	105 m		3 m	5.55	10.9 %
pdc	4,821	4,781	9.19	32.5		3 m	1 m	23.98	7.0 %
too_large	6961	6,958	6.06	17.67		5 m	1 m	13.11	3.3 %
b17s	39,390	39,360	97.92	187.59		110 m	3 m	180.74	4.7 %
b18s	107,979	107,957	110.48	177.54		12 h	5 m	167.64	4.8 %

*: from our machine, **: using a placer based on [7] on a 1.5 GHz Pentium CPU, with 2GB of memory.